APPENDIX C

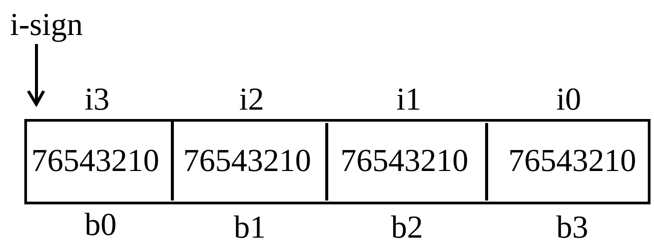
Internal Representation of Data Types

This appendix contains the detailed internal representations of the PDS standard data types listed in Table 3.2 of the Data Type Definitions chapter of this document.

C.1 MSB_INTEGER

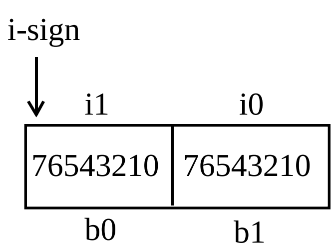
Aliases: INTEGER, MAC_INTEGER, SUN_INTEGER

MSB 4-byte integers:



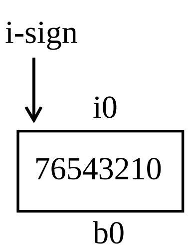
* Bit 7 in i3 is used for the sign bit.

MSB 2-byte integers:



* Bit 7 in i1 is used for the sign bit.

MSB 1-byte integers:



* Bit 7 is used for the sign bit.

Where:

b0 - b3 = Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3).

i-sign = integer sign bit

i0 - i3 = Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left, (e.g., lowest value =bit 0, highest value = bit 7) in the following way:

4-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-6 represent 2**24 through 2**30

2-bytes:

In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-6 represent 2**8 through 2**14

1-byte:

In i0, bits 0-6 represent 2**0 through 2**6

All negative signed values are assumed to be twos-compliment.

C.2 MSB_UNSIGNED_INTEGER

Aliases: MAC_UNSIGNED_INTEGER, SUN_UNSIGNED_INTEGER, UNSIGNED_INTEGER

MSB 4 byte unsigned integers:

i3	i2	i1	i0
76543210	76543210	76543210	76543210
b0	b1	b2	b3

MSB 2-byte unsigned integers:

i1	i0
76543210	76543210
b0	b1

MSB 1-byte unsigned integers:

i0
76543210
b0

Where:

b0 - b3 = Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3).

i0 - i3 = Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left, (e.g., lowest value =bit 0, highest value = bit 7) in the following way:

4-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-7 represent 2**24 through 2**31

2-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15

1-byte:
In i0, bits 0-7 represent 2**0 through 2**7

C.3 LSB_INTEGER

Aliases: PC_INTEGER, VAX_INTEGER

LSB 4-byte integers:

			i-sign
i0	i1	i2	i3
76543210	76543210	76543210	76543210
b0	b1	b2	b3

* Bit 7 in i3 is used for the sign bit.

LSB 2-byte integers:

	i-sign
i0	i1
76543210	76543210
b0	b1

* Bit 7 in i1 is used for the sign bit.

LSB 1-byte integers:

i-sign
i0
76543210
b0

* Bit 7 in i1 is used for the sign bit.

Where:

b0 - b3 = Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3).

i-sign = integer sign bit

i0 - i3 = Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left, (e.g., lowest value =

bit 0, highest value = bit 7) in the following way:

4-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-6 represent 2**24 through 2**30

2-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-6 represent 2**8 through 2**14

1-byte:
In i0, bits 0-6 represent 2**0 through 2**6

All negative signed values are assumed to be twos-compliment.

C.4 LSB_UNSIGNED_INTEGER

Aliases: PC_UNSIGNED_INTEGER, VAX_UNSIGNED_INTEGER

LSB 4-byte unsigned integers:

i0	i1	i2	i3
76543210	76543210	76543210	76543210
b0	b1	b2	b3

LSB 2-byte unsigned integers:

i0	i1
76543210	76543210
b0	b1

LSB 1-byte unsigned integers:

i0
76543210
b0

Where:

b0 - b3 = Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3).

i0 - 13 =Arrangement of bytes in the integer, from lowest order to highest order. The bits within each byte are interpreted from right to left, (e.g., lowest value =bit 0, highest value = bit 7) in the following way:

4-bytes:

In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15
In i2, bits 0-7 represent 2**16 through 2**23
In i3, bits 0-7 represent 2**24 through 2**31

2-bytes:
In i0, bits 0-7 represent 2**0 through 2**7
In i1, bits 0-7 represent 2**8 through 2**15

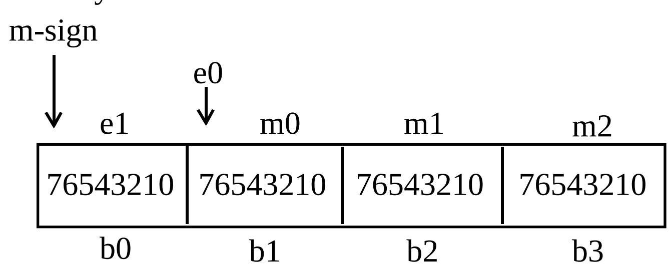
1-byte:

In i0, bits 0-7 represent 2**0 through 2**7

C.5 IEEE_REAL

Aliases: FLOAT, MAC_REAL, REAL, SUN_REAL

IEEE 4-byte real numbers:



IEEE 8-byte (double precision) real numbers:

m-sign

e1	e0 m0	m1	m2	m3	m4	m5	m6
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4	b5	b6	b7

* Bit 7 in e1 is used for the mantissa sign bit.

IEEE 10-byte (temporary) real numbers:

m-sign   int-bit (always 1)

e1	e0	m0	m1	m2
76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4

m3	m4	m5	m6	m7
76543210	76543210	76543210	76543210	76543210
b5	b6	b7	b8	b9

* Bit 7 in e1 is used for the mantissa sign bit.

Where:

b0 - b9 = Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, b3, etc.).

m-sign = Mantissa sign bit

int-bit = In l0 byte reals only, the implicit "1" is actually specified by this bit.

e0 - e1 = Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left, (e.g.,lowest value = rightmost bit in the exponent part of the byte, highest value = leftmost bit in the exponent part of the byte) in the following way:

4-bytes (single precision):

In e0, bit 7 represents 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 127

8-bytes (double precision):

In e0, bits 4-7 represent 2**0 through 2**3
In e1, bits 0-6 represent 2**4 through 2**10

Exponent bias = 1023

10-bytes (temporary):
In e0, bits 0-7 represent 2**0 through 2**7
In e1, bits 0-6 represent 2**8 through 2**14

Exponent bias = 16383

m0 - m7 = Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to the lowest order fractions. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

4 -bytes (single precision):
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23

8-bytes (double precision):
In m0, bits 3-0 represent 1/2**1 through 1/2**4
In m1, bits 7-0 represent 1/2**5 through 1/2**12
In m2, bits 7-0 represent 1/2**13 through 1/2**20
In m3, bits 7-0 represent 1/2**21 through 1/2**28
In m4, bits 7-0 represent 1/2**29 through 1/2**36
In m5, bits 7-0 represent 1/2**37 through 1/2**44
In m6, bits 7-0 represent 1/2**45 through 1/2**52

10-bytes (temporary):
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23
In m3, bits 7-0 represent 1/2**24 through 1/2**31
In m4, bits 7-0 represent 1/2**32 through 1/2**39
In m5, bits 7-0 represent 1/2**40 through 1/2**47
In m6, bits 7-0 represent 1/2**48 through 1/2**55
In m7, bits 7-0 represent 1/2**56 through 1/2**63

These representations all follow the format:

1. (mantissa) x 2** (exponent - bias)

with the "1." part implicit (except for the 10-byte temp real, in which the "1." part is actually stored in the third byte (b2)), In all cases, the exponent is stored as an unsigned, biased integer (e.g., exponent-as-stored - bias = true exponent value).

C.6 IEEE_COMPLEX

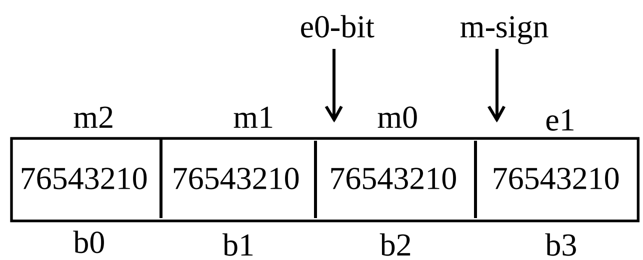
Aliases: COMPLEX, MAC_COMPLEX, SUN_COMPLEX

Two contiguous IEEE_REALs in memory, representing the real and imaginary parts.

C.7 PC_REAL

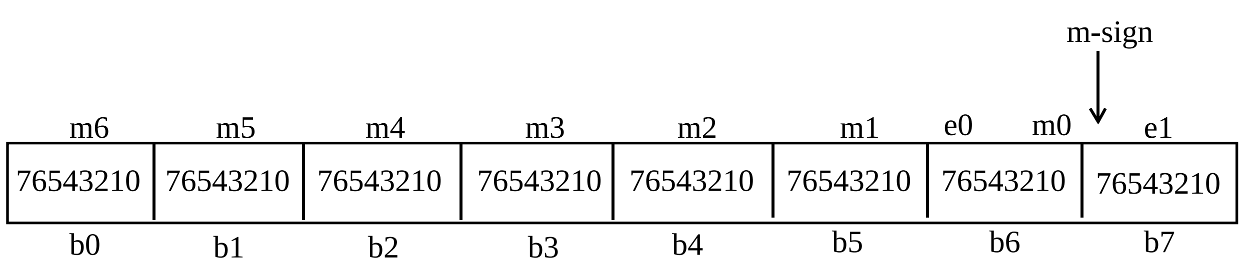
Aliases: None

PC 4-byte real numbers:



* Bit 7 in e1 is used for the mantissa sign bit.

PC 8-byte (double precision) real numbers:



* Bit 7 in e1 is used for the mantissa sign bit.

PC 10-byte (temporary) real numbers:

m7	m6	m5	m4	m3
76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4

		int-bit ↓		m-sign ↓
m2	m1	m0	e0	e1
76543210	76543210	76543210	76543210	76543210
b5	b6	b7	b8	b9

Where:

b0 - b9 = Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, b3, etc.).

m-sign = Mantissa sign bit

int-bit = In 10 byte reals only, the implicit "1" is actually specified by this bit.

e0 - e1 = Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left, (e.g., lowest value = rightmost bit in the exponent part of the byte, highest value = leftmost bit in the exponent part of the byte) in the following way:

4-bytes (single precision) :
In e0, bit 7 represents 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 127

8-bytes (double precision) :
In e0, bits 4-7 represent 2**0 through 2**3
In e1, bits 0-6 represent 2**4 through 2**10

Exponent bias = 1023

10-bytes (temporary):
In e0, bits 0-7 represent 2**0 through 2**7
In e1, bits 0-6 represent 2**4 through 2**10

Exponent bias = 16383

m0 - m7 = Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to lowest order fractions. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

4-bytes (single precision) :
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23

8-bytes (double precision) :
In m0, bits 3-0 represent 1/2**1 through 1/2**4
In m1, bits 7-0 represent 1/2**5 through 1/2**12
In m2, bits 7-0 represent 1/2**13 through 1/2**20
In m3, bits 7-0 represent 1/2**21 through 1/2**28
In m4, bits 7-0 represent 1/2**29 through 1/2**36
In m5, bits 7-0 represent 1/2**37 through 1/2**44
In m6, bits 7-0 represent 1/2**45 through 1/2**52

10-bytes (temporary) :
In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23
In m3, bits 7-0 represent 1/2**24 through 1/2**31
In m4, bits 7-0 represent 1/2**32 through 1/2**39
In m5, bits 7-0 represent 1/2**40 through 1/2**47
In m6, bits 7-0 represent 1/2**48 through 1/2**55
In m7, bits 7-0 represent 1/2**56 through 1/2**63

These representations all follow the format:

1. (mantissa) x 2**(exponent - bias)

with the "1." part implicit (except for the 10-byte temp real, in which the "1." part is actually stored in the third byte (b2)),

In all cases, the exponent is stored as an unsigned, biased integer (e.g., exponent-as-stored - bias=true exponent value).

C.8 PC_COMPLEX

Aliases: None

Two contiguous PC-REALs in memory, representing the real and imaginary parts.

C.9 VAX_REAL, VAXG_REAL

Aliases: VAX_DOUBLE (for VAX_REAL only, none for VAXG_REAL)

VAX F-type 4-byte real numbers:

e0 ↓	m-sign ↓		
m0	e1	m2	m1
76543210	76543210	76543210	76543210
b0	b1	b2	b3

* Bit 7 in e1 is used for the mantissa sign bit.

VAX D-type 8-byte real numbers:

e0-bit ↓	m-sign ↓						
m0	e1	m2	m1	m4	m3	m6	m5
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4	b5	b6	b7

* Bit 7 in e1 is used for the mantissa sign bit.

VAX G-type 8-byte real numbers:

	m-sign ↓						
e0 m0	e1	m2	m1	m4	m3	m6	m5
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4	b5	b6	b7

VAX H-type 16-byte real numbers:

m-sign

e0	e1	m1	m0	m3	m2	m5	m4
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b0	b1	b2	b3	b4	b5	b6	b7

m7	m6	m9	m8	m11	m10	m13	m12
76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
b8	b9	b10	b11	b12	b13	b14	b15

Where:

b0 - b15 = Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, b3, etc.).

m-sign = Mantissa sign bit

e0 - e1 = Arrangement of the portions of the bytes that make up the exponent, from lowest order to highest order. The bits within each byte are interpreted from right to left, (e.g., lowest value= rightmost bit in the exponent part of the byte, highest value = leftmost bit in the exponent part of the byte) in the following way:

4-bytes (F-type, single precision) :
In e0, bit 7 represents 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 129

8-bytes (D-type, double precision) :
In e0, bit 7 represents 2**0
In e1, bits 0-6 represent 2**1 through 2**7

Exponent bias = 129

8-bytes (G-type, double precision) :
In e0, bits 4-7 represent 2**0 through 2**3
In e1, bits 0-6 represent 2**4 through 2**10

Exponent bias = 1025

16-bytes (H-type) :

In e0, bits 0-7 represent 2**0 through 2**7
In e1, bits 0-6 represent 2**8 through 2**14

Exponent bias = 16385

m0 -m13 = Arrangement of the portions of the bytes that make up the mantissa, from highest order fractions to lowest order fractions. The order of the bits within each byte progresses from left to right, with each bit representing a fractional power of two, in the following way:

4-bytes (F-type, single precision) :

In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23

8-bytes (D-type, double precision) :

In m0, bits 6-0 represent 1/2**1 through 1/2**7
In m1, bits 7-0 represent 1/2**8 through 1/2**15
In m2, bits 7-0 represent 1/2**16 through 1/2**23
In m3, bits 7-0 represent 1/2**24 through 1/2**31
In m4, bits 7-0 represent 1/2**32 through 1/2**39
In m5, bits 7-0 represent 1/2**40 through 1/2**47
In m6, bits 7-0 represent 1/2**48 through 1/2**55

8-bytes (G-type, double precision) :

In m0, bits 3-0 represent 1/2**1 through 1/2**4
In m1, bits 7-0 represent 1/2**5 through 1/2**12
In m2, bits 7-0 represent 1/2**13 through 1/2**20
In m3, bits 7-0 represent 1/2**21 through 1/2**28
In m4, bits 7-0 represent 1/2**29 through 1/2**36
‘ In m5, bits 7-0 represent 1/2**37 through 1/2**44
In m6, bits 7-0 represent 1/2**45 through 1/2**52

16-bytes (H-type) :

In m0, bits 7-0 represent 1/2**1 through 1/2**8
In m1, bits 7-0 represent 1/2**9 through 1/2**16
In m2, bits 7-0 represent 1/2**17 through 1/2**24
In m3, bits 7-0 represent 1/2**25 through 1/2**32
In m4, bits 7-0 represent 1/2**33 through 1/2**40
In m5, bits 7-0 represent 1/2**41 through 1/2**48
In m6, bits 7-0 represent 1/2**49 through 1/2**56
In m7, bits 7-0 represent 1/2**57 through 1/2**64
In m8, bits 7-0 represent 1/2**65 through 1/2**72
In m9, bits 7-0 represent 1/2**73 through 1/2**80

In m10, bits 7-0 represent 1/2**81 through 1/2**88
In m11, bits 7-0 represent 1/2**89 through 1/2**96
In m12, bits 7-0 represent 1/2**97 through 1/2**104
In m13, bits 7-0 represent 1/2**105 through 1/2**112

These representations all follow the format:

1. (mantissa) x 2**(exponent - bias)

with the "1." part implicit

In all cases, the exponent is stored as an unsigned, biased integer (e.g., exponent-as-stored - bias = true exponent value).

C.10 VAX_COMPLEX, VAXG_COMPLEX

Aliases: None

Two contiguous VAX_REALs or VAXG_REALs in memory, representing the real and imaginary parts.

C.11 MSB_BIT_STRING

Aliases: BIT_STRING

MSB n-byte bit strings:

As read from a file:

bits 1-8	bits 9-16	bits 17-24	bits 25-32		bits ((nx8)-7) - (nx8)
76543210	76543210	76543210	76543210	. . .	
b0	b1	b2	b3		b x (n-1)

No byte swapping is needed.
Note: for n-byte bitstrings, continue pattern above.

MSB 2-byte bit strings:

As read from file:

bits 1-8	bits 9-16
76543210	76543210
b0	b1

No byte swapping is needed.

MSB 1-byte bit strings:

As read from file:

bits 1-8
76543210
b0

No byte swapping is needed.

Where:

b0 - b3 =Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3).

The bits within a byte are numbered from left to right:

```
76543210
↑      ↑
bit 1  bit 8
```

C.12 LSB_BIT_STRING

Aliases: VAX_BIT_STRING

LSB 4-byte bit strings:

As read from a file:

bits 25-32	bits 17-24	bits 9-16	bits 1-8
76543210	76543210	76543210	76543210
b0	b1	b2	b3

After bytes are swapped:

bits 1-8	bits 9-16	bits 17-24	bits 25-32
76543210	76543210	76543210	76543210
b3	b2	b1	b0

LSB 2-byte bit strings:

As read from a file:

bits 9-16	bits 1-8
76543210	76543210
b0	b1

After bytes are swapped:

bits 1-8	bits 9-16
76543210	76543210
b1	b0

LSB 1-byte bit strings:

As read from file:

bits 1-8
76543210
b0

No byte swapping is needed.

Where:

b0 - b3 =Arrangement of bytes as they appear when read from a file (e.g., read b0 first, then b1, b2, and b3).

The bits within a byte are numbered from left to right:

```
 76543210
 ↑      ↑
bit 1  bit 8
```